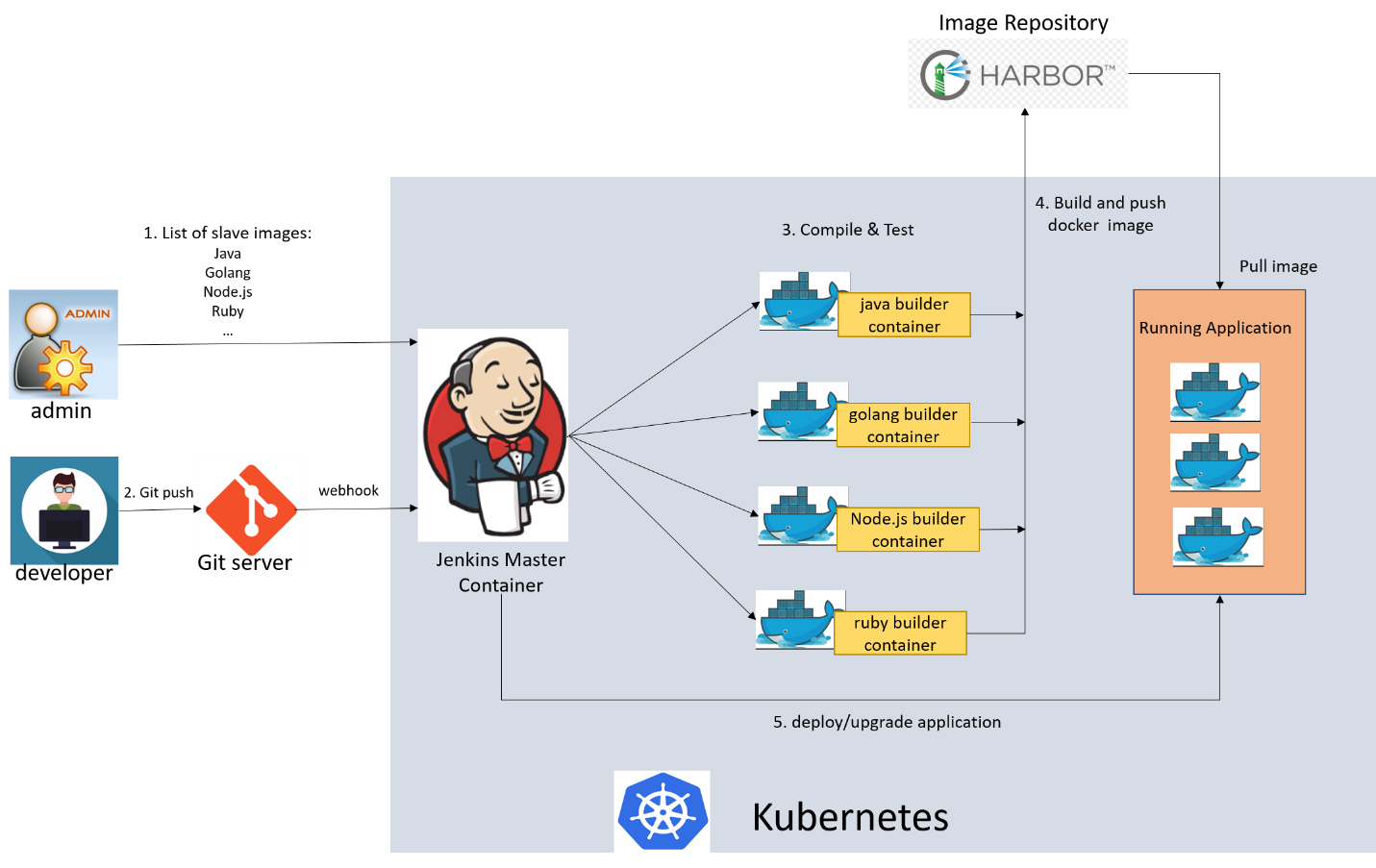
**ECP CI/CD Introduction**

ECP CI/CD brings seamless integration for the software development and deployment process, from the moment code is checked-in to the moment the service is updated with the latest code from that same check-in.

**Architecture Overview**



**Functionalities**

1. Automates the scaling of Jenkins slaves running in k8s.
2. Dynamically provision Jenkins slave, run a single build, then tear-down that slave.
3. Admin configures pod templates (e.g., golang, java, nodejs, …) to ensure a consistent build environment.
4. Users are allowed to select pod template to build their projects.
5. Integration with both public and private source code management: github, gitlab.
6. Integration with registry server to provide secure image management.
7. Build-in tools in Jenkins slave: docker, kubectl and helm.
8. Build-in gitlab and registry services in k8s.

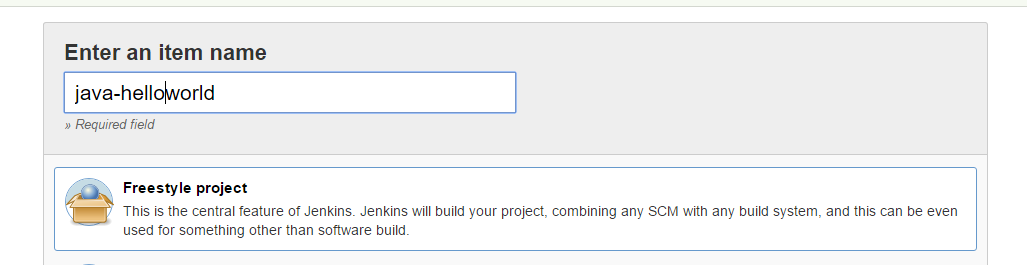
**Demo Case 1: Fresh build and deploy app**

1. Fork project to your github/gitlab: <https://github.com/maweina/docker-java-helloworld.git>
2. Login Jenkins: http:// 223.223.188.133:31091

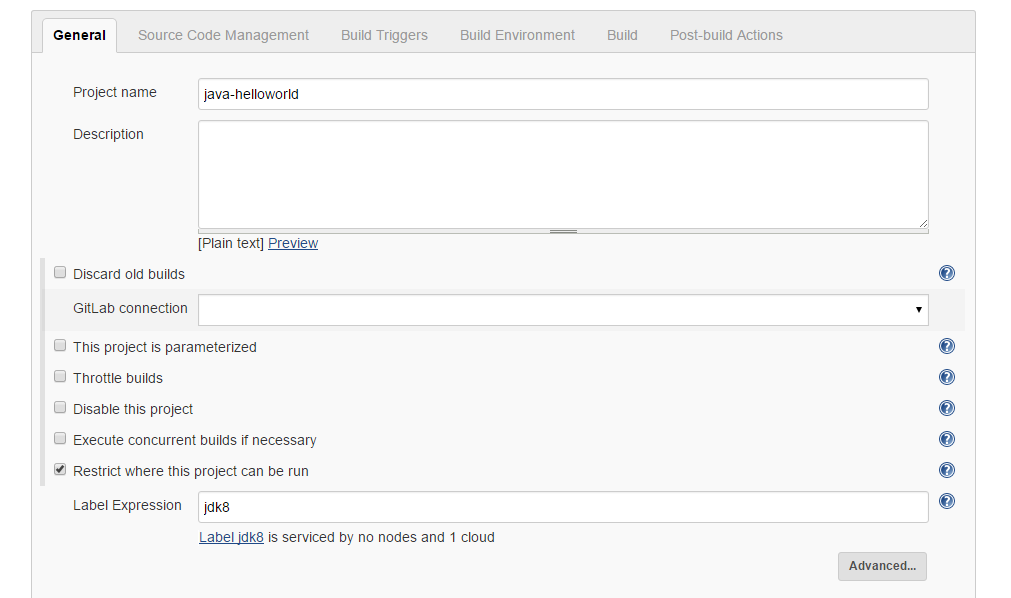
User name: admin

Password: admin

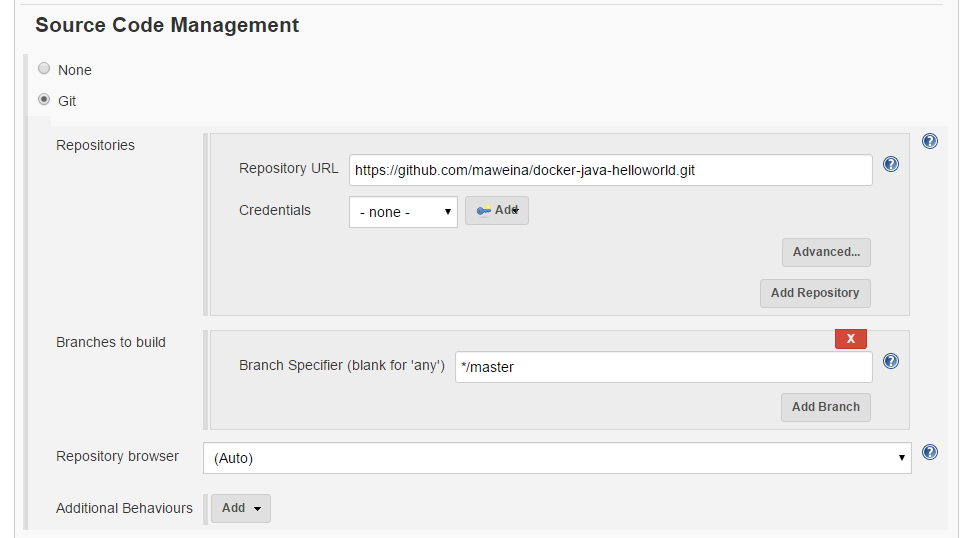
1. Create a freestyle project named “java-helloworld”



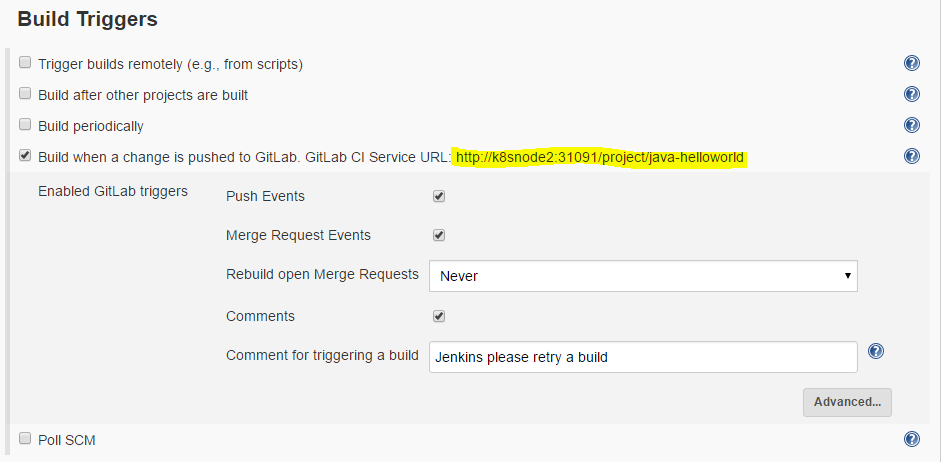
1. Configure project:
   1. Specify Jenkins slave node label to restrict this project can only run with jdk8 installed



* 1. Configure your github/gitlab repository URL, credential if applicable and branch



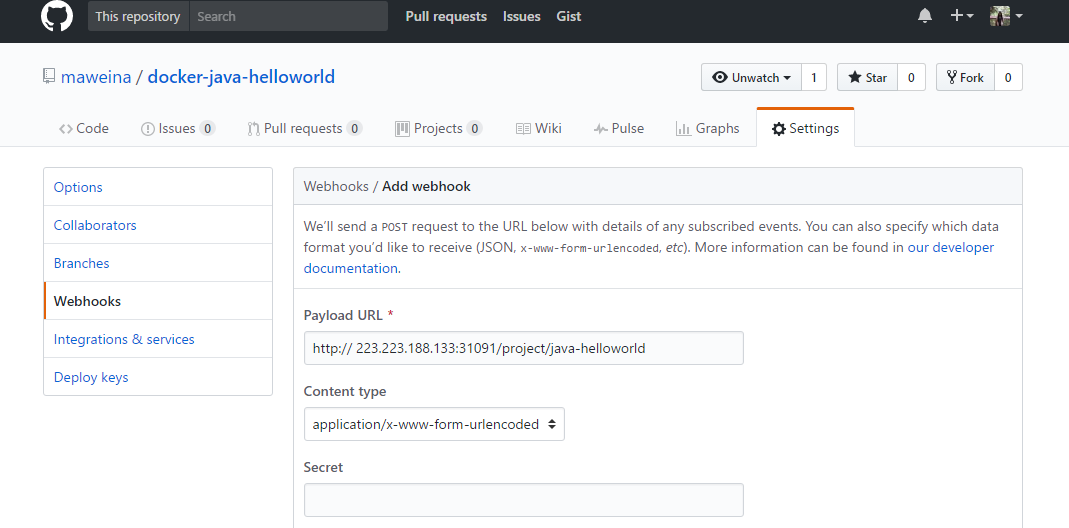
* 1. Tigger build when push code



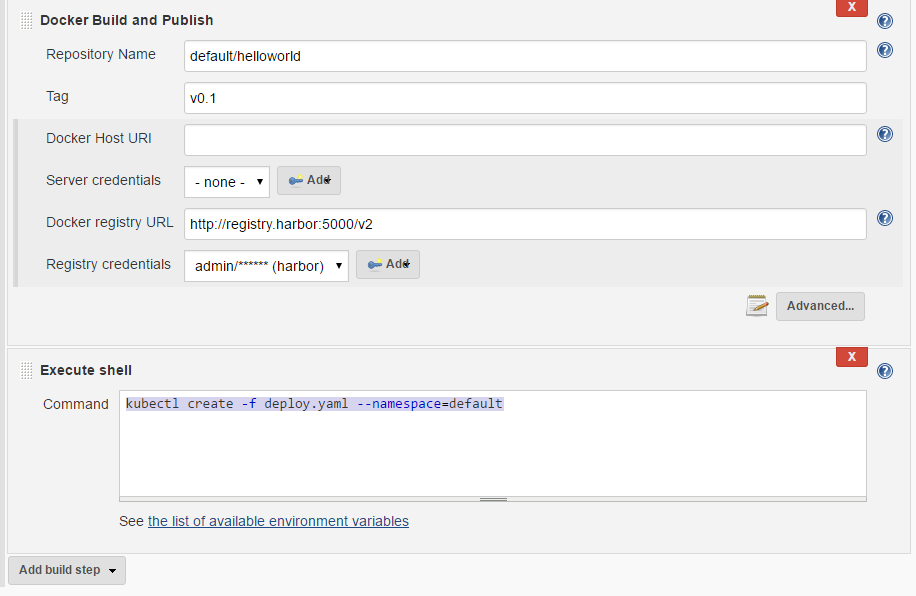
* 1. Copy the URL that is highlighted in above picture; from github/gitlab GUI, add a webhook. In our demo environment, the URL should be “http:// 223.223.188.133:31091/project/java-helloworld”

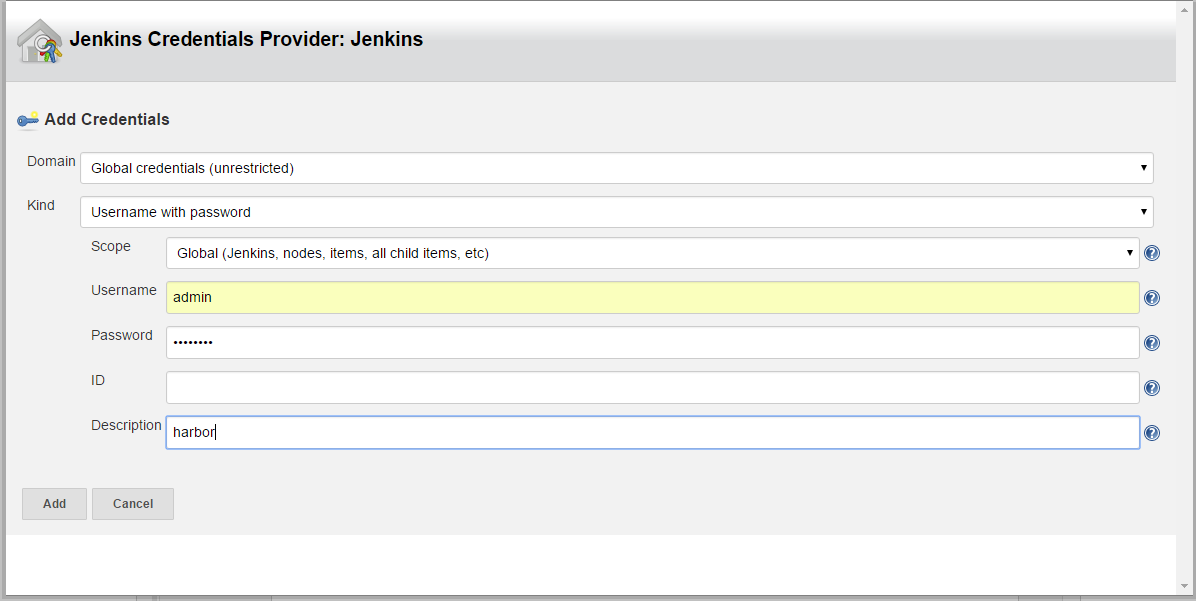
223.223.188.133:31091: jenkins public access url

java-helloworld: jenkins project name



* 1. Add three build steps:
     1. Excute shell: “javac helloworld.java”
     2. Docker Build and Publish
        + Repository Name: default/helloworld
        + Docker Registry URL: <http://registry.harbor:5000/v2>
        + Registry credentials: click “Add”; click “Jenkins”; type Username “admin”, password “password”, and Description “harbor”; select “harbor”
     3. Excute shell: “kubectl create -f deploy.yaml --namespace=default”

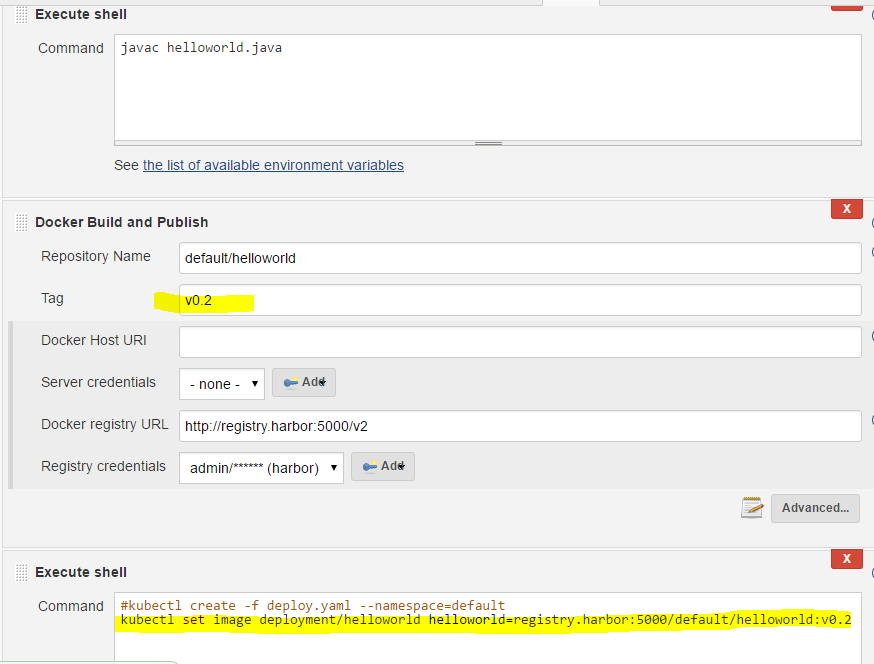




1. Automatically trigger a build by pushing code into gitlab/github. Or manually trigger a build from Jenkins.
2. Finally, you will see a new deployment “helloworld” under namespace “default” running.

**Demo Case 2: Upgrade app**

1. Configure Jenkins project
   1. Build a new image and tagged “v0.2”
   2. Rolling update “helloworld” using new image



1. Automatically trigger a build by pushing code into gitlab/github. Or manually trigger a build from Jenkins.
2. Finally, you will see deployment “helloworld” image is changed to “v0.2”.

